QSM368ZP-WF&SG368Z 系列 Linux&Ubuntu&OpenWrt 编译&烧录指导

智能产品

版本：1.2

日期：2025-11-26

状态：受控文件

上海移远通信技术股份有限公司（以下简称“移远通信”）始终以为客户提供最及时、最全面的服务为宗旨。如需任何帮助，请随时联系我司上海总部，联系方式如下：

上海移远通信技术股份有限公司
上海市闵行区田林路 1016 号科技绿洲 3 期（B 区）5 号楼　　邮编：200233
电话：+86 21 5108 6236　　邮箱：info@quectel.com

或联系我司当地办事处，详情请登录：https://www.quectel.com.cn/contact。

如需技术支持或反馈我司技术文档中的问题，请随时登录网址：
https://www.quectel.com.cn/contact?tab=t 或发送邮件至：support@quectel.com。

前言

移远通信提供该文档内容以支持客户的产品设计。客户须按照文档中提供的规范、参数来设计产品。同时，您理解并同意，移远通信提供的参考设计仅作为示例。您同意在设计您目标产品时使用您独立的分析、评估和判断。在使用本文档所指导的任何硬软件或服务之前，请仔细阅读本声明。您在此承认并同意，尽管移远通信采取了商业范围内的合理努力来提供尽可能好的体验，但本文档和其所涉及服务是在“可用”基础上提供给您的。移远通信可在未事先通知的情况下，自行决定随时增加、修改或重述本文档。

使用和披露限制

许可协议

除非移远通信特别授权，否则我司所提供硬软件、材料和文档的接收方须对接收的内容保密，不得将其用于除本项目的实施与开展以外的任何其他目的。

版权声明

移远通信产品和本协议项下的第三方产品可能包含受移远通信或第三方材料、硬软件和文档版权保护的相关资料。除非事先得到书面同意，否则您不得获取、使用、向第三方披露我司所提供的文档和信息，或对此类受版权保护的资料进行复制、转载、抄袭、出版、展示、翻译、分发、合并、修改，或创造其衍生作品。移远通信或第三方对受版权保护的资料拥有专有权，不授予或转让任何专利、版权、商标或服务商标权的许可。为避免歧义，除了正常的非独家、免版税的产品使用许可，任何形式的购买都不可被视为授予许可。对于任何违反保密义务、未经授权使用或以其他非法形式恶意使用所述文档和信息的违法侵权行为，移远通信有权追究法律责任。

商标

除另行规定，本文档中的任何内容均不授予在广告、宣传或其他方面使用移远通信或第三方的任何商标、商号及名称，或其缩略语，或其仿冒品的权利。

第三方权利

您理解本文档可能涉及一个或多个属于第三方的硬软件和文档（“第三方材料”）。您对此类第三方材料的使用应受本文档的所有限制和义务约束。

移远通信针对第三方材料不做任何明示或暗示的保证或陈述，包括但不限于任何暗示或法定的适销性或特定用途的适用性、平静受益权、系统集成、信息准确性以及与许可技术或被许可人使用许可技术相关的不侵犯任何第三方知识产权的保证。本协议中的任何内容都不构成移远通信对任何移远通信产品或任何其他硬软件、设备、工具、信息或产品的开发、增强、修改、分销、营销、销售、提供销售或以其他方式维持生产的陈述或保证。此外，移远通信免除因交易过程、使用或贸易而产生的任何和所有保证。

隐私声明

为实现移远通信产品功能，特定设备数据将会上传至移远通信或第三方服务器（包括运营商、芯片供应商或您指定的服务器）。移远通信严格遵守相关法律法规，仅为实现产品功能之目的或在适用法律允许的情况下保留、使用、披露或以其他方式处理相关数据。当您与第三方进行数据交互前，请自行了解其隐私保护和数据安全政策。

免责声明

1) 移远通信不承担任何因未能遵守有关操作或设计规范而造成损害的责任。
2) 移远通信不承担因本文档中的任何因不准确、遗漏、或使用本文档中的信息而产生的任何责任。
3) 移远通信尽力确保开发中功能的完整性、准确性、及时性，但不排除上述功能错误或遗漏的可能。除非另有协议规定，否则移远通信对开发中功能的使用不做任何暗示或法定的保证。在适用法律允许的最大范围内，移远通信不对任何因使用开发中功能而遭受的损害承担责任，无论此类损害是否可以预见。
4) 移远通信对第三方网站及第三方资源的信息、内容、广告、商业报价、产品、服务和材料的可访问性、安全性、准确性、可用性、合法性和完整性不承担任何法律责任。

文档历史

修订记录

版本	日期	作者	变更描述
-	2024-07-12	Szymon Yu	文档创建
1.0	2024-07-18	Szymon Yu	受控版本
1.1	2025-02-26	Ivan Zujovic	1. 新增 Ubuntu 编译环境版本及备注（第 2.1 章）。 2. 新增 OpenWrt 操作系统的编译及烧录相关说明。 3. 新增关于 FTP（ftp://202.111.194.162）服务器的用途备注（第 2.2.1 章）。 4. 新增 OpenWrt 操作系统下的开发示例（第 4.3 章）。
1.2	2025-11-26	Damian Li	1. 代码块增加换行符提示。 2. 新增关于项目开发注意事项和 PDF 复制代码可能导致非预期换行的说明备注（第 1 章）。 3. 新增 Kernel 6.1 编译命令相关内容（第 3.2 章）。

目录

表格索引

图片索引

1 引言

本文档主要用于指导客户如何在移远通信 QSM368ZP-WF 产品和 SG368Z 系列模块上获取、编译和加载智能产品的软件，包括：

- 建立开发环境并安装相关工具；
- 编译 Linux、Ubuntu 或 OpenWrt 代码，并将生成的固件包烧录至设备中。

备注

1. 本文档适用于运行 Linux 操作系统的 QSM368ZP-WF 产品和运行 Linux 或 Ubuntu 或 OpenWrt 操作系统的 SG368Z 系列模块。
2. SG368Z 系列模块本身无法预装 OpenWrt 系统，仅提供 SDK 及开发指导供客户二次开发。若有问题，请联系移远通信技术支持。
3. 项目开发注意事项说明，可从 FTP（ftp://202.111.194.162，路径：*/Common*）获取《智能模块客户软件开发注意事项》进行参考。
4. FTP（ftp://202.111.194.162）用于存放工具、驱动和镜像等相关资料供用户下载使用，无其他用途。
5. 由于 PDF 格式特性，文档中部分代码块可能因页面宽度限制自动换行。直接从 PDF 复制代码到编辑器时，可能引入非预期的换行符（\n），导致代码无法正常运行。

   建议操作：

   若需从 PDF 复制，粘贴后请注意完成如下动作：
   - 手动删除复制粘贴后代码中的多余换行符。
   - 或通过代码编辑器的替换功能批量删除多余换行符（注意保留语义换行）。

2 开发环境

2.1. 编译环境

表 1：编译环境

描述	版本	作用
Linux 开发工作站的性能高于在一般桌面系统上运行 Ubuntu 64 位操作系统的最低要求。为加快编译时间，建议使用性能强大的 PC，该 PC 应当符合如下规格： **CPU**：Intel i7-2600 @ 3.4 GHz 或以上 **内存**：16 GB RAM 或以上 **硬盘**：200 GB 或以上（首选 SSD 加速）	Ubuntu 64 位 18.04 LTS/ Ubuntu 64 位 20.04 LTS/ Ubuntu 64 位 22.04 LTS	Linux 编译机
Windows 系统	Windows 10 或更高版本	烧录固件包

备注

1. 推荐使用 Ubuntu 64 位 18.04 LTS（用于编译 Linux 系统）或 64 位 20.04 LTS（用于编译 Ubuntu 系统）或 64 位 22.04 LTS（用于编译 OpenWrt 系统）。移远通信已经验证了 Ubuntu 64 位 18.04 LTS、Ubuntu 64 位 20.04 LTS 和 Ubuntu 64 位 22.04 LTS 版本的编译工作，对于其他 Ubuntu 版本，移远通信暂未验证。
2. 编译前需要获取相关代码，请联系移远通信技术支持获取 GitLab 访问权限来下载 SDK。

2.2. 安装和启动工具

2.2.1. 在 Windows 上安装工具

1. RKDevTool

   从 FTP（ftp://202.111.194.162，路径：*Linux/tools/windows/RKDevTool_Release_v2.92.zip*、*Ubuntu/tools/windows/RKDevTool_Release_v2.92.zip* 或 *OpenWRT/tools/windows/RKDevTool_Release_v2.92.zip*）下载 RKDevTool 并安装。

2. USB 驱动

   从 FTP（ftp://202.111.194.162，路径：*Linux/tools/windows/DriverAssitant_v5.11.zip*、*Ubuntu/tools/windows/DriverAssitant_v5.11.zip* 或 *OpenWRT/tools/windows/DriverAssitant_v5.11.zip*）下载 USB 驱动并安装。

2.2.2. 在 Ubuntu 上安装工具

若使用的 Ubuntu 版本满足***第 2.1 章***的编译环境要求，可以执行如下命令在 Ubuntu 上安装相关工具，然后参考***第 3 章***进行编译和烧录步骤。

```
sudo apt-get install repo git ssh make gcc libssl-dev liblz4-tool expect g++ \
patchelf chrpath gawk texinfo chrpath diffstat binfmt-support qemu-user-static \
live-build bison flex fakeroot cmake    gcc-multilib g++-multilib unzip \
device-tree-compiler python-pip libncurses5-dev zstd rsync device-tree-compiler bc time
```

备注

若编译主机默认的系统 shell 是 Dash 而非 Bash，请按照如下步骤将符号链接**/bin/sh→dash** 更改为**/bin/sh→bash**：

1. 执行 **sudo rm /bin/sh** 命令删除符号链接**/bin/sh**；
2. 执行 **sudo ln -s /bin/bash /bin/sh** 命令，创建一个指向**/bin/bash** 的符号链接，并命名为**/bin/sh**。

2.2.3. 在 Ubuntu 上使用 Docker

若使用的 Ubuntu 版本不满足***第 2.1 章***的编译环境要求，可以在进行***第 3 章***的编译和烧录步骤前，执行如下步骤使用 Docker 环境。

2.2.3.1. 安装 Docker

如下以 Ubuntu 16.04 主机为例介绍 Docker 安装步骤。

1. 执行如下命令更新 APT 软件包索引：

   ```
   $sudo apt-get update
   ```

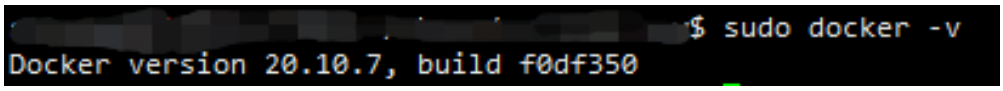
2. 执行如下命令在 Ubuntu 上安装 Docker：

   ```
   $sudo apt install docker.io
   ```

3. 执行如下命令验证 Docker 是否安装成功：

```
$sudo docker -v
```

若 Docker 安装成功，执行结果如下图所示：



2.2.3.2. 将 Ubuntu 18.04/20.04/22.04 镜像导入 Docker

1. 从 FTP（ftp://202.111.194.162/，路径：*Linux/Tools/*）下载 *smartlinux-docker-ubuntu1804_v2.tar*（Linux 系统）或从 FTP（ftp://202.111.194.162/，路径：*Ubuntu/Tools/*）下载 *smartlinux-rk3568-ubuntu-20.04-v1.3.tar.gz*（Ubuntu 系统）或从 FTP（ftp://202.111.194.162/，路径：*OpenWRT/Tools/*）下载 *smartlinux-rk3568-ubuntu-22.04-v1.0.tar.gz*（OpenWrt 系统）。

2. 执行如下命令将镜像导入 Docker：

   - Linux 系统：

     ```
     $ sudo docker load -i smartlinux-docker-ubuntu1804_v2.tar
     ```

   - Ubuntu 系统：

     ```
     $ sudo docker load -i smartlinux-rk3568-ubuntu-20.04-v1.3.tar.gz
     ```

   - OpenWrt 系统：

     ```
     $ sudo docker load -i smartlinux-rk3568-ubuntu-22.04-v1.0.tar.gz
     ```

3. 执行如下命令查看导入到 Docker 的镜像：

```
$ sudo docker images
```

导入的 Docker 镜像如下图所示：

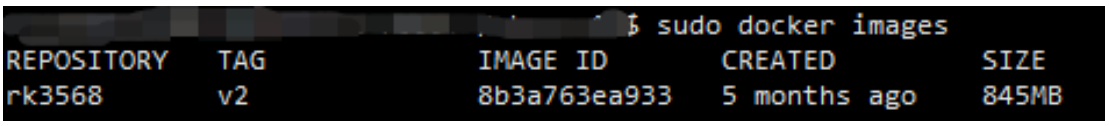


图 1：导入到 Docker 的镜像（Linux 系统）

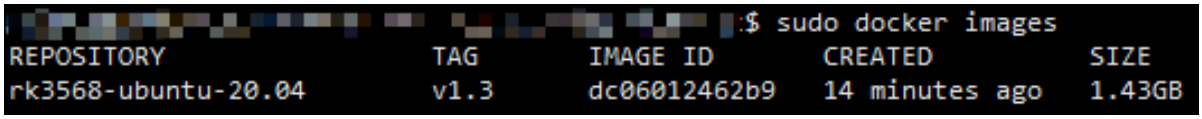


图 2：导入到 Docker 的镜像（Ubuntu 系统）

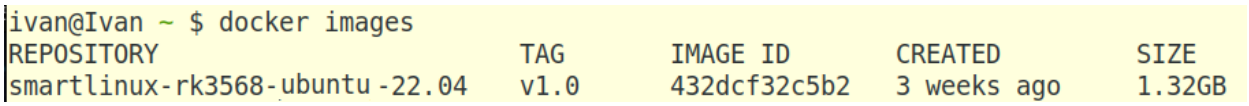


图 3：导入到 Docker 的镜像（OpenWrt 系统）

4. 执行如下命令运行镜像（推荐参考***备注 2***）：

   - Linux 系统：

     ```
     $ sudo docker run -it 8b3a763ea933 /bin/bash
     ```

   - Ubuntu 系统：

     ```
     $ sudo docker run --privileged -it dc06012462b9 /bin/bash
     ```

   - OpenWrt 系统：

     ```
     $ sudo docker run --privileged -it 432dcf32c5b2 /bin/bash
     ```

备注

1. 默认已在 Docker 镜像中创建了用户名为“quectel”的账户。
2. 若 PC 编译机的根目录下空间不足（空闲存储 200 GB），可以将主机中需要的目录映射到 Docker 镜像中。例如，执行如下命令，将主机中的*/home/quectel* 目录映射到 Docker 镜像。

   - Linux 系统：

     ```
     $ cd /home/quectel/
     $ mkdir share
     $ sudo docker run -it -v /home/quectel/share:/home/test/share 8b3a763ea933 /bin/bash
     ```

   - Ubuntu 系统：

     ```
     $ cd /home/quectel/
     $ mkdir share
     ```

```
$ sudo docker run --privileged -it -v /home/quectel/share:/home/test/share \
dc06012462b9 /bin/bash
```

- OpenWrt 系统：

```
$ cd /home/quectel/
$ cd /home/quectel/
$ mkdir share
$ sudo docker run --privileged -it -v /home/quectel/share:/home/test/share \
432dcf32c5b2 /bin/bash
```

5. 使用“**Ctrl**”+“**P**”和“**Ctrl**”+“**Q**”退出 Docker 镜像。

6. 执行如下命令查看 Docker 容器 ID：

```
$ sudo docker ps -a
```

Docker 容器 ID 如下图所示：

```
$ sudo docker ps -a
CONTAINER ID   IMAGE          COMMAND       CREATED              STATUS                          PORTS     NAMES
4443cc11518c   8b3a763ea933   "/bin/bash"   About a minute ago   Exited (0) 13 seconds ago                 bold_einstein
```

图 4：Docker 容器 ID（Linux 系统）

```
$ sudo docker ps -a
CONTAINER ID   IMAGE          COMMAND       CREATED              STATUS                          PORTS     NAMES
e9d987f699de   dc06012462b9   "/bin/bash"   About a minute ago   Exited (0) 10 seconds ago                 pedantic_meitner
```

图 5：Docker 容器 ID（Ubuntu 系统）

```
ivan@Ivan /media/magacin/docker/Images $ docker ps
CONTAINER ID   IMAGE          COMMAND             CREATED         STATUS         PORTS     NAMES
264b70f7c860   432dcf32c5b2   "/bin/bash"         3 minutes ago   Up 2 minutes             reverent_lovelace
```

图 6：Docker 容器 ID（OpenWrt 系统）

7. 执行如下命令再次进入 Docker 容器：

```
$ sudo docker attach CONTAINER ID
```

8. 执行如下命令停用 Docker 容器：

```
$ sudo docker stop CONTAINER ID
```

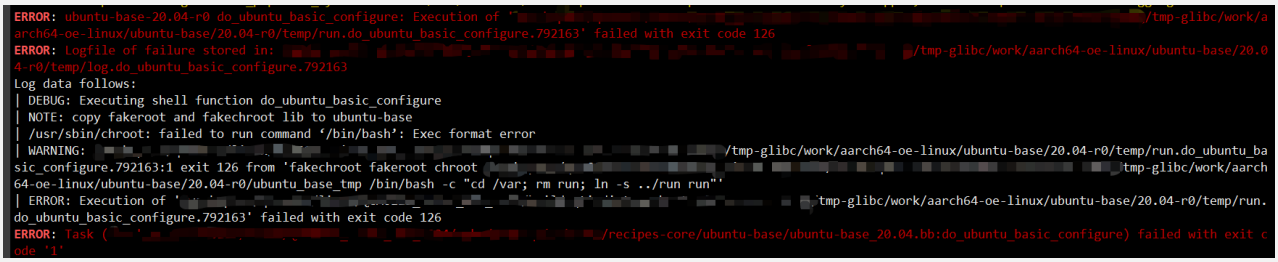
9. 若需要重启 Docker 容器，执行如下命令启用 Docker 容器：

```
$ sudo docker start CONTAINER ID
```

备注

1. Docker 镜像上已安装所有工具，用户无需进行任何操作。
2. Docker 镜像默认有两个用户：
   - 用户名：root
   - 用户名：quectel

   用户需要添加一个自己的新用户进行编译。例如，通过 **useradd test** 添加名为“test”的用户。登录 Docker bash 时，可以使用 **su test** 切换到“test”用户来构建 SDK。
3. 若在 Ubuntu 系统上编译时报错如下：



此时，在 Docker 之外的主机上，从 http://archive.ubuntu.com/ubuntu/pool/universe/q/qemu/下载最新的“qemu-user-static”deb 包并安装，如下所示：

```
$ wget http://archive.ubuntu.com/ubuntu/pool/universe/q/qemu/\
qemu-user-static_6.2+dfsg-2ubuntu6.12_amd64.deb
$ sudo dpkg -i qemu-user-static_6.2+dfsg-2ubuntu6.12_amd64.deb
```

3 编译与烧录

3.1. 访问 GitLab 克隆代码

1. 联系移远通信技术支持获取 GitLab 账号；
2. 登录 GitLab（https://git-master.quectel.com/）；
3. 将 SSH 公钥添加到 GitLab 服务器；
4. 在 shell 中键入以下命令：
   - Linux 系统：

     ```
     git clone "ssh://git@git-master.quectel.com:8407/smart/rk3568_linux_r60_v1.3.2.git"
     ```

   - Ubuntu 系统：

     ```
     git clone "ssh://git@git-master.quectel.com:8407/smart/rk3568_ubuntu_r60_v1.3.2.git"
     ```

   - OpenWrt 系统：

     ```
     git clone "ssh://git@git-master.quectel.com:8407/smart/rk3568_openwrt_r61_v1.4.0.git"
     ```

详细 GitLab 操作可参考***文档 [1]***。

3.2. 编译命令

3.2.1. Linux 系统

3.2.1.1. Kernel 5.10 及之前

1. 执行如下命令进行整编：

```
$ cd RK3568_Linux_R60_v1.3.2
$ source build-quec.sh
$ build-all-image
```

编译完成后，生成的镜像位于 *rockdev/*路径下。

2. 执行如下命令清除所有内容：

```
$ cd RK3568_Linux_R60_v1.3.2
$ source build-quec.sh
$ buildclean
```

3. 执行如下命令编译 uboot：

```
$ cd RK3568_Linux_R60_v1.3.2
$ source build-quec.sh
$ makeuboot
```

编译完成后，生成的 *uboot.img* 位于 *u-boot/*路径下。

4. 执行如下命令编译 kernel：

```
$ cd RK3568_Linux_R60_v1.3.2
$ source build-quec.sh
$ makekernel
```

编译完成后，生成的 *boot.img* 位于 *kernel/*路径下。

5. 执行如下命令编译 recovery：

```
$ cd RK3568_Linux_R60_v1.3.2
$ source build-quec.sh
$ makerecovery
```

编译完成后，生成的 *recovery.img* 位于 *buildroot/output/rockchip_rk356x_recovery/images/*路径下。

6. 执行如下命令编译 rootfs：

```
$ cd RK3568_Linux_R60_v1.3.2
$ source build-quec.sh
$ makerootfs-buildroot
```

编译完成后，生成的 *rootfs.ext2* 位于 *buildroot/output/rockchip_rk3568/images/*路径下。

3.2.1.2. Kernel 6.1

1. 执行如下命令进行整编：

```
$ cd RK3568_Linux6.1_R62_rkr5
$ source build-quec.sh
$ build-all-image
```

编译完成后，生成的镜像位于 *rockdev/*路径下。

2. 执行如下命令清除所有内容：

```
$ cd RK3568_Linux6.1_R62_rkr5
$ source build-quec.sh
$ buildclean
```

3. 执行如下命令编译 uboot：

```
$ cd RK3568_Linux6.1_R62_rkr5
$ source build-quec.sh
$ makeuboot
```

编译完成后，生成的 *uboot.img* 位于 *u-boot/*路径下。

4. 执行如下命令编译 kernel：

```
$ cd RK3568_Linux6.1_R62_rkr5
$ source build-quec.sh
$ makekernel
```

编译完成后，生成的 *boot.img* 位于 *kernel/*路径下。

5. 执行如下命令编译 recovery：

```
$ cd RK3568_Linux6.1_R62_rkr5
$ source build-quec.sh
$ makerecovery
```

编译完成后，生成的 *recovery.img* 位于 *buildroot/output/rockchip_rk356x_recovery/images/*路径下。

6. 执行如下命令编译 rootfs：

```
$ cd RK3568_Linux6.1_R62_rkr5
$ source build-quec.sh
$ makerootfs-buildroot
```

编译完成后，生成的 *rootfs.ext2* 位于 *buildroot/output/rockchip_rk3568/images/*路径下。

3.2.2. Ubuntu 系统

1. 执行如下命令进行整编：

```
$ cd rk3568_ubuntu_r60_v1.3.2
$ source build-quec.sh
$ build-all-image-yocto
```

编译完成后，生成的镜像位于 *rockdev/*路径下。

2. 执行如下命令清除所有内容：

```
$ cd rk3568_ubuntu_r60_v1.3.2
$ source build-quec.sh
$ buildclean
```

3. 执行如下命令编译 uboot：

```
$ cd rk3568_ubuntu_r60_v1.3.2
$ source build-quec.sh
$ makeuboot
```

编译完成后，生成的 *uboot.img* 位于 *u-boot/*路径下。

4. 执行如下命令编译 kernel：

```
$ cd rk3568_ubuntu_r60_v1.3.2
$ source build-quec.sh
$ makekernel
```

编译完成后，生成的 *boot.img* 位于 *kernel/*路径下。

5. 执行如下命令编译 recovery：

```
$ cd rk3568_ubuntu_r60_v1.3.2
$ source build-quec.sh
```

```
$ makerecovery
```

编译完成后，生成的 *recovery.img* 位于 *buildroot/output/rockchip_rk356x_recovery/images/*路径下。

6. 执行如下命令编译 rootfs：

```
$ cd rk3568_ubuntu_r60_v1.3.2
$ source build-quec.sh
$ makerootfs-yocto
```

编译完成后，生成的 *qti-ubuntu-robotics-image-rockchip-rk3568-evb.rootfs.ext4* 位于 *yocto/build/tmp-glibc/deploy/images/rockchip-rk3568-evb/*路径下。

3.2.3. OpenWrt 系统

1. 执行如下命令进行整编：

```
$ cd rk3568_openwrt_r61_v1.4.0
$ source build-quec.sh
$ build-all-image-openwrt
```

编译完成后，生成的镜像位于 *rockdev/*路径下。

2. 执行如下命令清除所有内容：

```
$ cd rk3568_openwrt_r61_v1.4.0
$ source build-quec.sh
$ buildclean
```

3. 执行如下命令编译 uboot：

```
$ cd rk3568_openwrt_r61_v1.4.0
$ source build-quec.sh
$ makeuboot
```

编译完成后，生成的 *uboot.img* 位于 *u-boot/*路径下。

4. 执行如下命令编译 kernel：

```
$ cd rk3568_openwrt_r61_v1.4.0
$ source build-quec.sh
$ makekernel
```

编译完成后，生成的 *boot.img* 位于 *kernel/*路径下。

5. 执行如下命令编译 recovery：

```
$ cd rk3568_openwrt_r61_v1.4.0
$ source build-quec.sh
$ makerecovery
```

编译完成后，生成的 *recovery.img* 位于 *buildroot/output/rockchip_rk356x_recovery/images/*路径下。

6. 执行如下命令编译 rootfs：

```
$ cd rk3568_openwrt_r61_v1.4.0
$ source build-quec.sh
$ makerootfs-buildroot
```

编译完成后，生成的 *rootfs.ext2* 位于 *buildroot/output/rockchip_rk3568/images/*路径下。

备注

1. 由于首次编译 SDK 时会从网上下载很多软件包，因此编译时间较长。
2. 为节省后续编译用时，可以将下载包保存到计算机上的公共文件夹中，然后更改 *buildroot/Config.in* 中 *BR2_DL_DIR* 的默认路径为公共文件夹路径。若是在 Ubuntu 系统上编译，还需更改 *yocto/build/conf/include/common.conf* 中 *DL_DIR* 的默认路径为公共文件夹路径；若是在 OpenWrt 系统上编译，还需更改 *openwrt/config/sg368z_defconfig* 中 *CONFIG_DOWNLOAD_FOLDER* 和 *buildroot/Config.in* 中 *BR2_DL_DIR* 的默认路径为公共文件夹路径。
3. 若从 GitLab 上更新 SDK，建议在编译之前按照上述***步骤 2*** 清除旧 SDK 包中的所有内容。

3.3. 构建、加载及烧录固件包

1. 从 FTP（ftp://202.111.162，路径：*Linux/tools/windows/*、*Ubuntu/tools/windows/*或 *OpenWRT/tools/windows/*）下载 *RKDevTool_Release_v2.92.zip* 软件包。

2. 将 Linux、Ubuntu 或 OpenWrt 镜像文件（*rockdev/*目录下编译生成的镜像）复制到 *RKDevTool_Release/*目录下，包含的文件如下图所示：

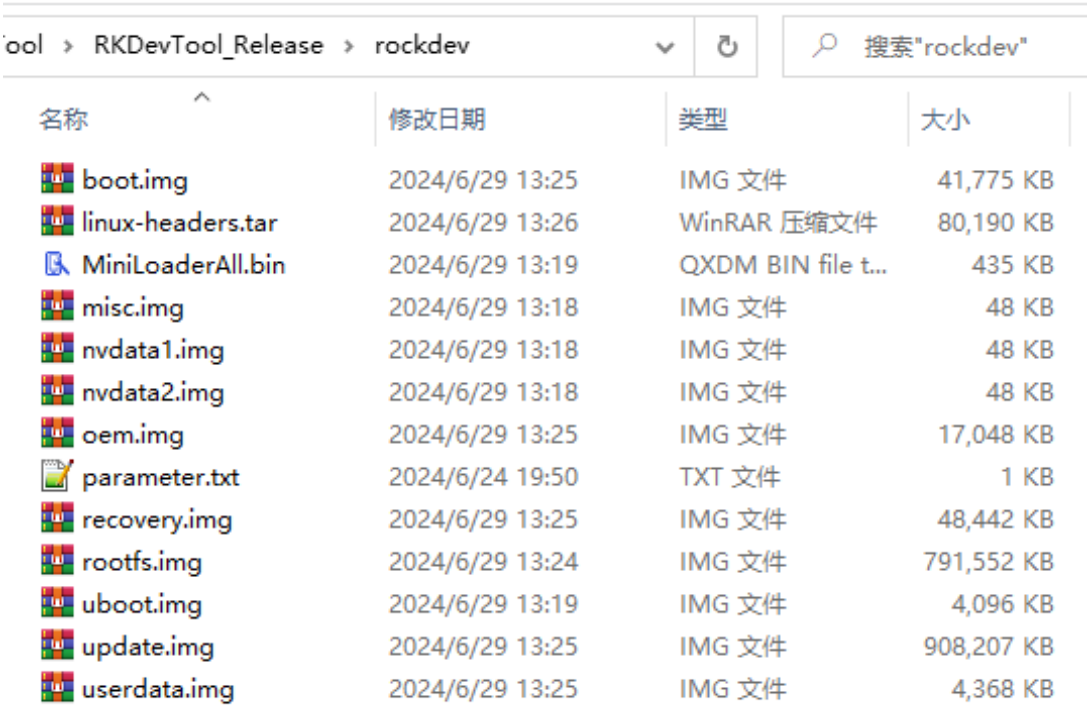


图 7：镜像文件

3. 进入“**LOADER**”下载模式：将 EVB 开发板上的“**BOOT**”开关拨动至“**ON**”，如下图所示。然后，启动 EVB 并给设备上电，设备将进入 USB Loader 状态。

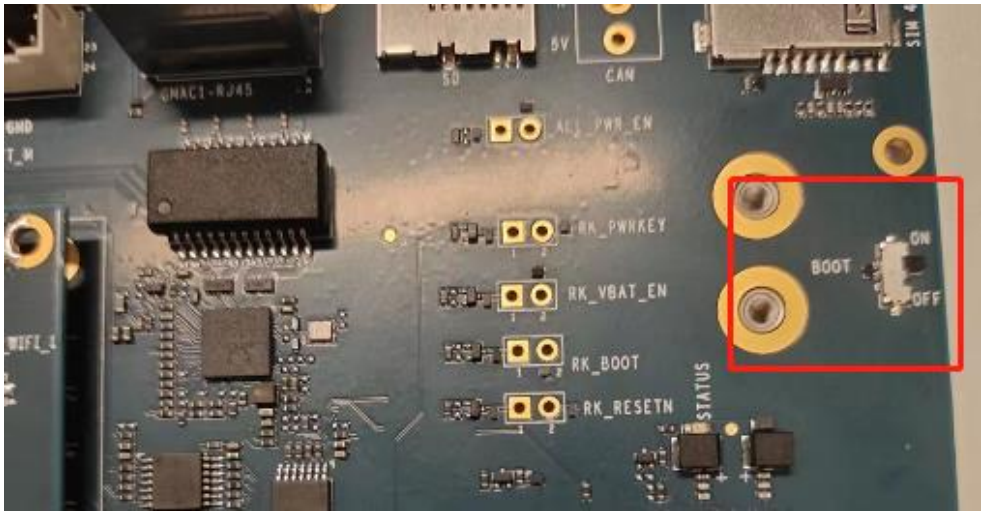


图 8：BOOT 开关

4. 运行 RKDevTool 工具，在工具中的分区表空白处单击右键，在弹出的对话框中选择“**导入配置**”，选择 *RKDevTool_Release/rockdev/parameter.txt* 文件，选择后将自动加载设备的分区配置信息，左键分别单击各分区表格最右侧按键（下图中的红框处），根据分区名称选择对应镜像文件。

图 9：固件选择

备注

1. *persist* 分区保存 Wi-Fi/BT 相关射频参数，禁止选择编译生成的 *persist.img* 进行烧录。
2. *nvdata1* 和 *nvdata2* 分区保存出厂参数，禁止选择编译生成的 *nvdata1.img* 和 *nvdata2.img* 进行烧录。

5. 点击“**执行**”按钮，等待固件包烧录完成：

```
测试设备开始
测试设备成功
正在下载Gpt...(100%)
开始下载uboot...
正在下载 uboot...(100%)
开始下载misc...
正在下载 misc...(100%)
开始下载boot...
正在下载 boot...(100%)
开始下载recovery...
正在下载 recovery...(100%)
开始下载rootfs...
正在下载 rootfs...(100%)
开始下载oem...
正在下载 oem...(100%)
开始下载userdata...
正在下载 userdata...(100%)
下载完成
```

图 10：烧录结果

6. 固件包烧录完成后，设备将自动重启。为了能正常进入设备系统，设备重启前，需要手动将 EVB 开发板上的“**BOOT**”开关拨动至“**OFF**”。

备注

设备的分区表文件路径为 *device/rockchip/rk356x/parameter-buildroot-fit.txt*。请勿自行修改默认分区，如需修改，请联系移远通信技术支持。

4 开发示例

4.1. Linux 系统下的开发示例

进入代码顶层目录，在 shell 中执行如下 **source** 命令：

```
$ source build-quec.sh
```

执行如下命令单独编译.mk 文件：

```
$ envsetup_buildroot
```

1. 通过 *CMakeLists* 编译 *helloworld*

   - 源码：*external/quectel-app/helloworld/*
   - .mk 文件配置：*buildroot/package/quectel-app/helloworld/helloworld.mk*
   - *Config.in* 文件配置：*buildroot/package/quectel-app/helloworld/Config.in*

   执行如下命令编译 *helloworld*：

   ```
   $ remake helloworld
   ```

   编译完成后，生成的文件位于 *buildroot/output/rockchip_rk3568/build/helloworld-1.0.0/*目录下。

2. 通过 *Autoconf* 编译 *apptest*

   - 源码：*external/quectel-core/apptest*
   - .mk 文件配置：*buildroot/package/quectel-core/apptest/apptest.mk*
   - *Config.in* 文件配置：*buildroot/package/quectel-core/apptest/Config.in*

执行如下命令编译 *apptest*：

```
$ remake apptest
```

编译完成后，生成的文件位于 *buildroot/output/rockchip_rk3568/build/apptest-1.0.0/*目录下。

3. 默认安装 *helloworld* 和 *apptest* 到系统

在 *buildroot/configs/quectel/quectel_core.config* 和 *buildroot/configs/quectel/quectel_app.config* 文件中打开编译宏，如下图所示：

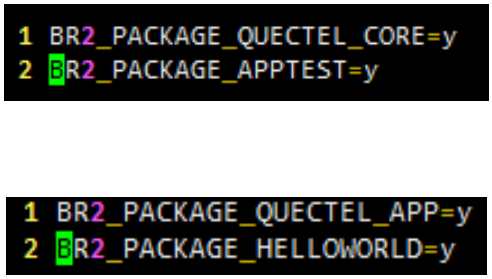


图 11：打开编译宏

此时，执行如下命令编译所有内容：

```
$ build-all-image
```

备注

有关 Buildroot 的更多信息，请访问链接 https://buildroot.org/。

4.2. Ubuntu 系统下的开发示例

进入代码顶层目录，在 shell 中执行如下 **source** 命令：

```
$ source build-quec.sh
```

执行如下命令单独编译.bb 文件：

```
$ envsetup_yocto
```

1. 通过 *CMakeLists* 编译 *helloworld*

   - 源码：*external/quectel-app/example/helloworld/*
   - .bb 文件配置：*yocto/meta-quectel-app/recipes/example/helloworld_0.1.bb*

   执行如下命令编译 *helloworld*：

   ```
   $ rebake helloworld
   ```

   编译完成后，生成的文件位于 *build/tmp-glibc/work/aarch64-oe-linux/helloworld/0.1-r0/*目录下。

2. 通过 *Autoconf* 编译 *apptest*

   - 源码：*external/quectel-app/example/apptest/*
   - .bb 文件配置：*yocto/meta-quectel-app/recipes/example/apptest_0.1.bb*

   执行如下命令编译 *apptest*：

   ```
   $ rebake apptest
   ```

   编译完成后，生成的文件位于 *build/tmp-glibc/work/rockchip_rk3568_evb-oe-linux/apptest/0.1-r0/* 目录下。

3. 默认安装 *helloworld* 和 *apptest* 到系统

   取消 *yocto/meta-quectel-app/recipes/packagegroups/packagegroup-quectel-app.bb* 红框内代码的注释，将 *helloworld* 和 *apptest* 安装到系统，如下图所示：

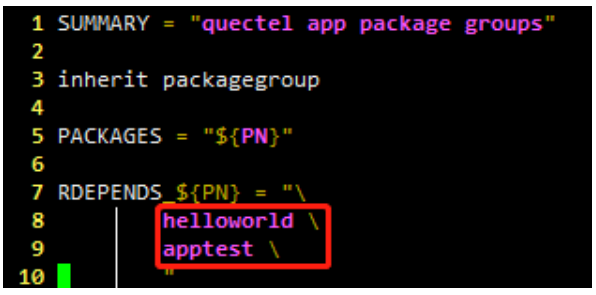


图 12：示例

此时，执行如下命令编译所有内容：

```
$ build-all-image-yocto
```

备注

有关 Yocto 详细信息，请访问链接 https://www.yoctoproject.org/。

4.3. OpenWrt 系统下的开发示例

进入代码顶层目录，在 shell 中执行如下 **source** 命令：

```
$ source build-quec.sh
```

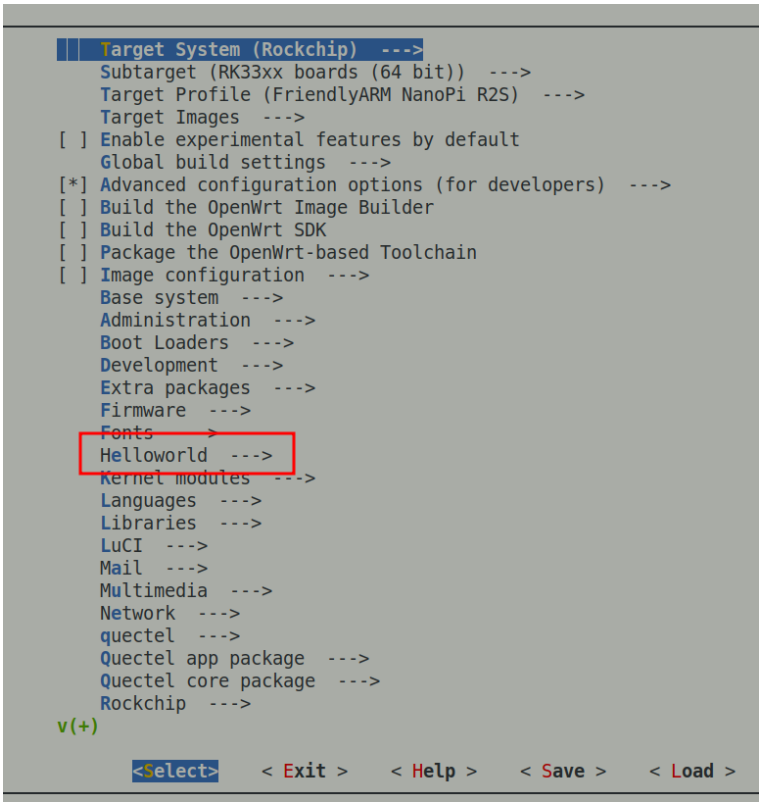
1. 通过 *Makefile* 编译 *helloworld*

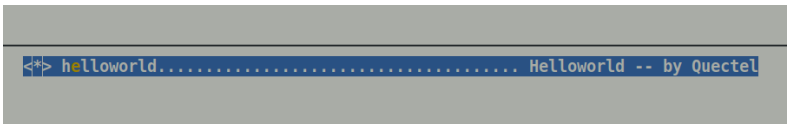
   - 源码：*external/quectel-app/helloworld/src/*
   - *Makefile* 文件配置：*openwrt/package/quectel-app/helloworld/Makefile*

   进入 *openwrt/*目录，执行如下命令进入配置界面：

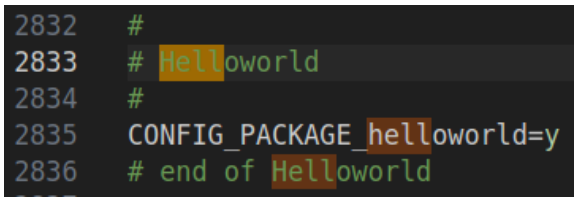
   ```
   make menuconfig
   ```

   配置界面如下，找到 *Helloworld* 包并按“**Enter**”键进行选择，如下图所示：

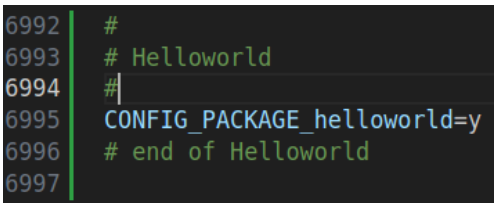




选择 *Helloworld* 包后，*openwrt/.config* 文件将会更新，保存修改后的配置并返回代码顶层目录，如下图所示：



将修改内容添加到 *sg368z_defconfig* 文件中，如下图所示：



此时，执行如下命令编译所有内容：

```
$ build-all-image-openwrt
```

编译完成后，检查二进制文件是否已成功编译，如下图所示：

备注

有关 Buildroot 详细信息，请访问链接 https://buildroot.org/。

5 附录 参考文档及术语缩写

表 2：参考文档

文档名称
[1] Quectel_智能模块_GitLab_用户指导

表 3：术语缩写

缩写	英文全称	中文全称
BT	Bluetooth	蓝牙
CPU	Central Processing Unit	中央处理器
EVB	Evaluation Board	评估板
FTP	File Transfer Protocol	文件传输协议
ID	Identifier	标识符
LTS	Transport Layer Security	传输层安全（协议）
OS	Operating System	操作系统
PC	Personal Computer	个人计算机
RF	Radio Frequency	射频
SDK	Software Development Kit	软件开发工具包
SSD	Solid State Disk/Drive	固态硬盘/驱动器
SSH	Secure Shell	安全外壳协议
USB	Universal Serial Bus	通用串行总线